INSTITUTO FEDERAL DO ACRE

CAMPUS RIO BRANCO

CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS PARA INTERNET

DISCIPLINA: SISTEMAS OPERACIONAIS

Discente: Sávio Henrique V. Alves

Professor: Me. Jonas Pontes

Conteúdo: Gerenciamento de memória e prática de Linux

Atividade: Análise do uso de memória no Linux

Instruções Gerais

Esta atividade tem como objetivo aprofundar os conhecimentos sobre gerenciamento de memória no Linux, combinando teoria e prática. Tu deves responder às questões propostas e realizar um experimento prático de análise de uso de memória.

Ao final, o aluno deve entregar um documento em formato *Portable Document Format* (PDF), contendo:
- Respostas às questões teóricas.
- Descrição e resultados do experimento prático.
- Análise dos resultados obtidos.

A atividade pode ser feita com pesquisa na internet, uso de ferramentas de inteligência artificial e consulta à documentação oficial do Linux.

Parte 1: Questões Teóricas

1. Explique a diferença entre memória RAM, memória *swap* e *buffers/cache*.

> **Memória RAM (Random Access Memory):** É a memória física de acesso aleatório, onde os dados são armazenados temporariamente para que o processador possa acessá-los rapidamente. É volátil, ou seja, perde seus dados quando o sistema é desligado.
> **Memória Swap:** É uma extensão da memória RAM que reside no disco rígido (HDD ou SSD). Quando a RAM está cheia, o sistema operacional move páginas de memória que não estão em uso ativo para o espaço de swap, liberando RAM para processos mais urgentes. O swap é mais lento que a RAM, pois depende da velocidade do disco.
> **Buffers:** São áreas de memória usadas para armazenar dados temporariamente enquanto estão sendo transferidos entre dispositivos (por exemplo, entre a RAM e o disco).
> **Cache:** É uma área de memória que armazena cópias de dados frequentemente acessados para acelerar o acesso futuro. O cache pode ser de disco, de rede, etc.

2. O que é paginação e qual sua relação com a memória virtual?

Paginação: É uma técnica de gerenciamento de memória que divide a memória física (RAM) e a memória virtual (espaço no disco) em blocos de tamanho fixo chamados "páginas". Quando a RAM está cheia, o sistema operacional move páginas menos usadas para o espaço de swap no disco, liberando espaço na RAM para novos dados.

Relação com a memória virtual: A memória virtual é uma abstração que permite que o sistema operacional use o disco como uma extensão da RAM. A paginação é o mecanismo que gerencia essa extensão, movendo páginas entre a RAM e o disco conforme necessário.

3. Como o Linux decide quando usar swap? Qual é o efeito da configuração vm.swappiness?

Uso de Swap: O Linux começa a usar swap quando a memória RAM está quase cheia e o sistema precisa liberar espaço para novos processos. O kernel do Linux monitora o uso de memória e decide quando mover páginas de memória para o swap.

vm.swappiness: É um parâmetro do kernel que controla a tendência do sistema em usar o swap.

O valor pode variar de 0 a 100:

0: O sistema evita ao máximo o uso de swap, usando a RAM até o limite.

100: O sistema usa o swap agressivamente, mesmo que haja RAM disponível.

Valor padrão: Geralmente é 60, o que significa que o sistema começa a usar swap quando a RAM está cerca de 40% cheia.

4. Qual é a função do comando free -m? Explique cada uma das colunas apresentadas na saída desse comando.

O comando **free -m** exibe informações sobre o uso de memória RAM e swap no sistema, com os valores em megabytes (MB).

Colunas:

total: Quantidade total de memória física (RAM) ou swap disponível.

used: Quantidade de memória que está sendo usada.

free: Quantidade de memória que está livre (não utilizada).

shared: Memória compartilhada entre processos (geralmente usada por programas que compartilham bibliotecas).

buff/cache: Memória usada para buffers e cache.

available: Estimativa de quanta memória está disponível para iniciar novos aplicativos sem usar swap.

5. Explique o funcionamento do comando vmstat e como ele pode ser útil para analisar o uso de memória.

Funcionamento: O comando vmstat (virtual memory statistics) exibe estatísticas sobre o uso de memória virtual, processos, E/S de disco, e atividade da CPU.

Utilidade: É útil para analisar o desempenho do sistema, identificando gargalos de memória, uso de CPU, e atividade de disco. Por exemplo:

Uso de memória: Mostra quanta memória está sendo usada, livre, e em swap.

Atividade de CPU: Mostra o tempo gasto em operações de usuário, sistema, e tempo ocioso.

E/S de disco: Mostra leituras e escritas por segundo.

6. Como o Linux gerencia a memória compartilhada entre processos? Dê exemplos de situações em que esse tipo de memória é vantajoso.

Gerenciamento de memória compartilhada: O Linux usa técnicas como **memória compartilhada (shared memory)** e **copy-on-write (COW)** para permitir que múltiplos processos acessem a mesma região de memória.

Memória Compartilhada: Regiões de memória são mapeadas para múltiplos processos, permitindo que eles compartilhem dados diretamente. Isso é comum em sistemas de interprocess communication (IPC).

Copy-on-Write (COW): Quando um processo tenta modificar uma página de memória compartilhada, o sistema cria uma cópia privada dessa página para o processo, mantendo a original intacta para outros processos.

Exemplos de situações:

Banco de dados: Várias instâncias de um banco de dados podem compartilhar a mesma memória para acessar tabelas ou índices comuns.

Servidores web: Múltiplos processos de um servidor web podem compartilhar

bibliotecas comuns em memória, reduzindo o uso de RAM.

Aplicativos com threads: Threads de um mesmo processo compartilham a mesma memória, o que permite comunicação rápida entre elas.

Parte 2: Prática no Linux

Nesta parte, tu deves executar um experimento para monitorar e analisar o uso da memória.

1. Coleta de Dados sobre o Uso de Memória

Execute os seguintes comandos no Linux e registre as saídas obtidas:

- free -m
- vmstat 1 10
- cat /proc/meminfo | grep -E 'MemTotal|MemFree|Buffers|Cached|SwapTotal|SwapFree'

- Apresente as saídas coletadas e explique o que elas significam.

```
mint@mint:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            3868        1809         164         283        2459        2058
Swap:              0           0           0
mint@mint:~$
```

O comando exibe o uso de memória RAM e swap em megabytes (MB).

total: Quantidade total de memória RAM ou swap disponível.

used: Quantidade de memória em uso.

free: Quantidade de memória livre.

shared: Memória compartilhada entre processos.

buff/cache: Memória usada para buffers e cache.

available: Estimativa de memória disponível para novos processos sem usar swap.

```
mint@mint:~$ vmstat 1 10
procs -----------memory---------- ---swap-- -----io---- -system-- -------cpu-------
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st gu
 0  0      0 231260   2716 2515536    0    0  1108     0  284    1  3  3 94  0  0  0
 0  0      0 231260   2716 2515520    0    0     0     0  431  584  2  1 97  0  0  0
 0  0      0 231260   2716 2515520    0    0     0     0  900 1282  3  2 95  0  0  0
 0  0      0 231260   2716 2515520    0    0     0     0  882 1187  4  2 94  0  0  0
 1  0      0 231260   2716 2515520    0    0     0     0  512  739  3  2 95  0  0  0
 0  0      0 235724   2716 2515552    0    0     0     0 1891 2131 16 10 74  0  0  0
 0  0      0 235980   2716 2515552    0    0     0     0 1284 1573 13  5 82  0  0  0
 1  0      0 236108   2716 2515552    0    0     0     0 1339 1766  7  6 87  0  0  0
 0  0      0 236108   2716 2515552    0    0     0     0 1232 1639  7  7 86  0  0  0
 0  0      0 236108   2716 2515552    0    0     0     0  487  641  3  2 96  0  0  0
mint@mint:~$
```

Este comando exibe estatísticas sobre o sistema, incluindo uso de memória, CPU, e E/S de disco, a cada 1 segundo, por 10 iterações.

memory:

swpd: Quantidade de memória swap usada (em KB).

free: Memória livre (em KB).

buff: Memória usada para buffers (em KB).

cache: Memória usada para cache (em KB).

swap:

si: Quantidade de memória movida do swap para a RAM por segundo (swap in).

so: Quantidade de memória movida da RAM para o swap por segundo (swap out).

cpu:

us: Tempo gasto em operações de usuário.

sy: Tempo gasto em operações do sistema.

id: Tempo ocioso (idle).

```
mint@mint:~$ cat /proc/meminfo | grep -E 'MemTotal|MemFree|Buffers|Cached|SwapTo
tal|SwapFree'
MemTotal:        3961024 kB
MemFree:          236800 kB
Buffers:            1084 kB
Cached:          2710056 kB
SwapCached:            0 kB
SwapTotal:             0 kB
SwapFree:              0 kB
mint@mint:~$
```

Permite visualizar detalhes da RAM e do Swap.

MemTotal: 3961024 kB

Total de memória RAM disponível no sistema: 3.961.024 kB (aproximadamente 3,78 GB).

MemFree: 236800 kB

Memória RAM livre: 236.800 kB (aproximadamente 231 MB).

Buffers: 1084 kB

Memória usada para buffers: 1.084 kB (aproximadamente 1 MB).

Cached: 2710056 kB

Memória usada para cache: 2.710.056 kB (aproximadamente 2,58 GB).

SwapCached: 0 kB

Memória em swap que foi recuperada para a RAM: 0 kB (nenhuma memória swap está sendo usada no momento).

SwapTotal: 0 kB

Total de espaço de swap disponível: 0 kB (nenhum espaço de swap está configurado no sistema).

SwapFree: 0 kB

Espaço de swap livre: 0 kB (como não há swap configurado, este valor também é zero).

2. Experimento de Sobrecarregamento de Memória

Crie um programa em uma linguagem de programação para alocar memória progressivamente e forçar o uso de swap. Use um tamanho que seja capaz de estourar a quantidade de memória. Insira o código usado e explique-o

```python
import time
import numpy as np

memory_blocks = []
allocation_size = 900_000_000  # Aloca  858 MB por segundo
duration = 20 # Executa por 20 segundos

for second in range(1, duration + 1):
    try:
        memory_blocks.append(np.ones((allocation_size,), dtype=np.uint8))
        allocated_gb = (second * allocation_size) / (1024 ** 3)
        print(f"Tempo {second}s - Memória alocada: {allocated_gb:.2f} GB")

    except MemoryError:
        print("Memória insuficiente, atingiu o limite de alocação.")
        break

    time.sleep(1)
```

O meu notebook tem 12 GB de memoria ram com vm.swappiness=60 (Padrão), então aloquei 858 MB de mémoria ram por segundo durante 20 segundos. Assim, forçando o uso de RAM até esgotar a memória disponível e forçar o uso de swap.

Durante a execução do programa. Tu podes usar o comando watch -n 1 free -m para tal fim.

Preferencialmente, crie um script Linux para gerenciar e monitorar todo o processo.

```
#!/bin/bash
echo "Timestamp,MemUsed,SwapUsed" > memory_usage.csv
python3 memory_overload.py & # Inicia o programa em Python em segundo plano
PYTHON_PID=$!
for i in {1..20}; do
 TIMESTAMP=$(date +"%T")
 MEM_USED=$(free -m | awk 'NR==2{print $3}')
 SWAP_USED=$(free -m | awk 'NR==3{print $3}')
 echo "$TIMESTAMP,$MEM_USED,$SWAP_USED" >> memory_usage.csv
 sleep 1
done
wait $PYTHON_PID
```

Aumente a o tamanho da área de swap e para além da demanda do teu código. Repita os testes.

Gere gráficos dois testes, quanto ao uso da memória e de swap, e o adicione como resposta a esta atividade.

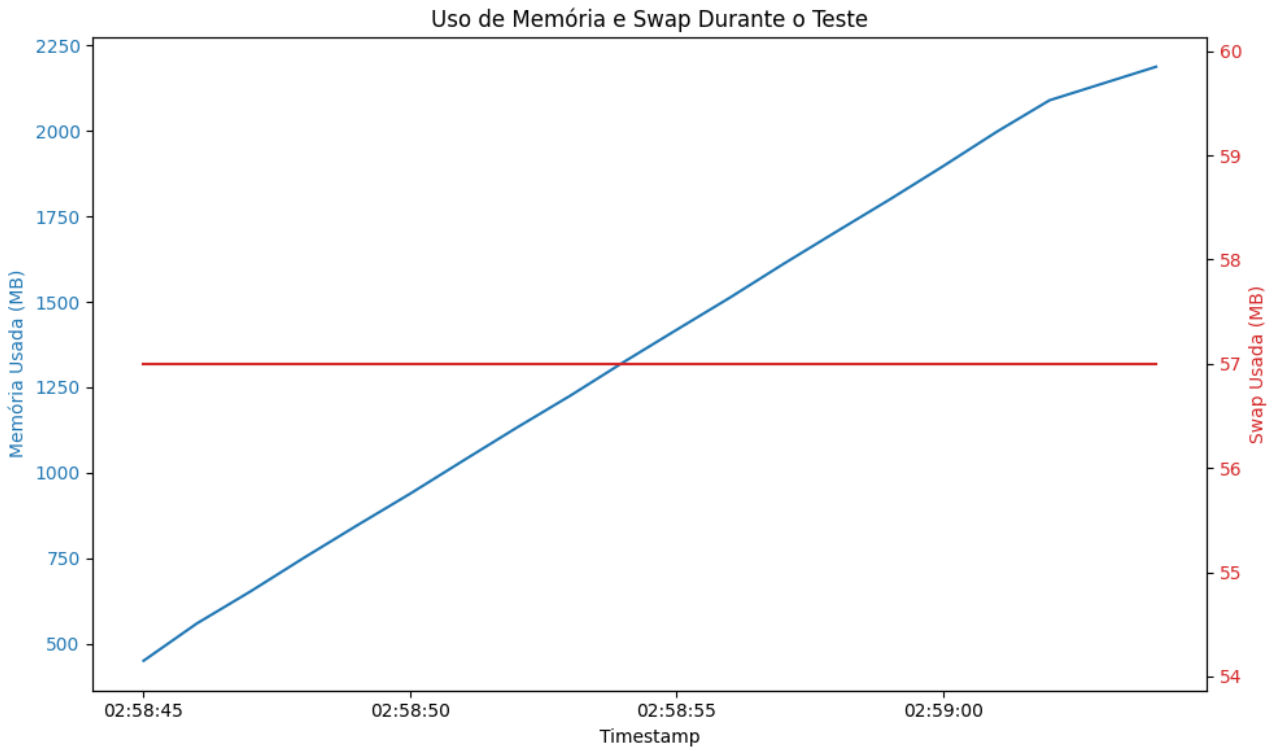
Teste 1 – Usando 100 MB/s durante 20s

```
(myenv) savioalves@Alves:~$ ./monitor_memory.sh
Tempo 1s - Memória alocada: 0.09 GB
Tempo 2s - Memória alocada: 0.19 GB
Tempo 3s - Memória alocada: 0.28 GB
Tempo 4s - Memória alocada: 0.37 GB
Tempo 5s - Memória alocada: 0.47 GB
Tempo 6s - Memória alocada: 0.56 GB
Tempo 7s - Memória alocada: 0.65 GB
Tempo 8s - Memória alocada: 0.75 GB
Tempo 9s - Memória alocada: 0.84 GB
Tempo 10s - Memória alocada: 0.93 GB
Tempo 11s - Memória alocada: 1.02 GB
Tempo 12s - Memória alocada: 1.12 GB
Tempo 13s - Memória alocada: 1.21 GB
Tempo 14s - Memória alocada: 1.30 GB
Tempo 15s - Memória alocada: 1.40 GB
Tempo 16s - Memória alocada: 1.49 GB
Tempo 17s - Memória alocada: 1.58 GB
Tempo 18s - Memória alocada: 1.68 GB
Tempo 19s - Memória alocada: 1.77 GB
Tempo 20s - Memória alocada: 1.86 GB
(myenv) savioalves@Alves:~$
```

Relátorio do Arquivo CSV

```
(myenv) savioalves@Alves:~$ cat memory_usage.csv
Timestamp,MemUsed,SwapUsed
02:58:45,451,57
02:58:46,560,57
02:58:47,653,57
02:58:48,751,57
02:58:49,846,57
02:58:50,939,57
02:58:51,1036,57
02:58:52,1132,57
02:58:53,1225,57
02:58:54,1323,57
02:58:55,1418,57
02:58:56,1512,57
02:58:57,1610,57
02:58:58,1705,57
02:58:59,1799,57
02:59:00,1896,57
02:59:01,1996,57
02:59:02,2089,57
02:59:03,2138,57
02:59:04,2187,57
```

Gráfico



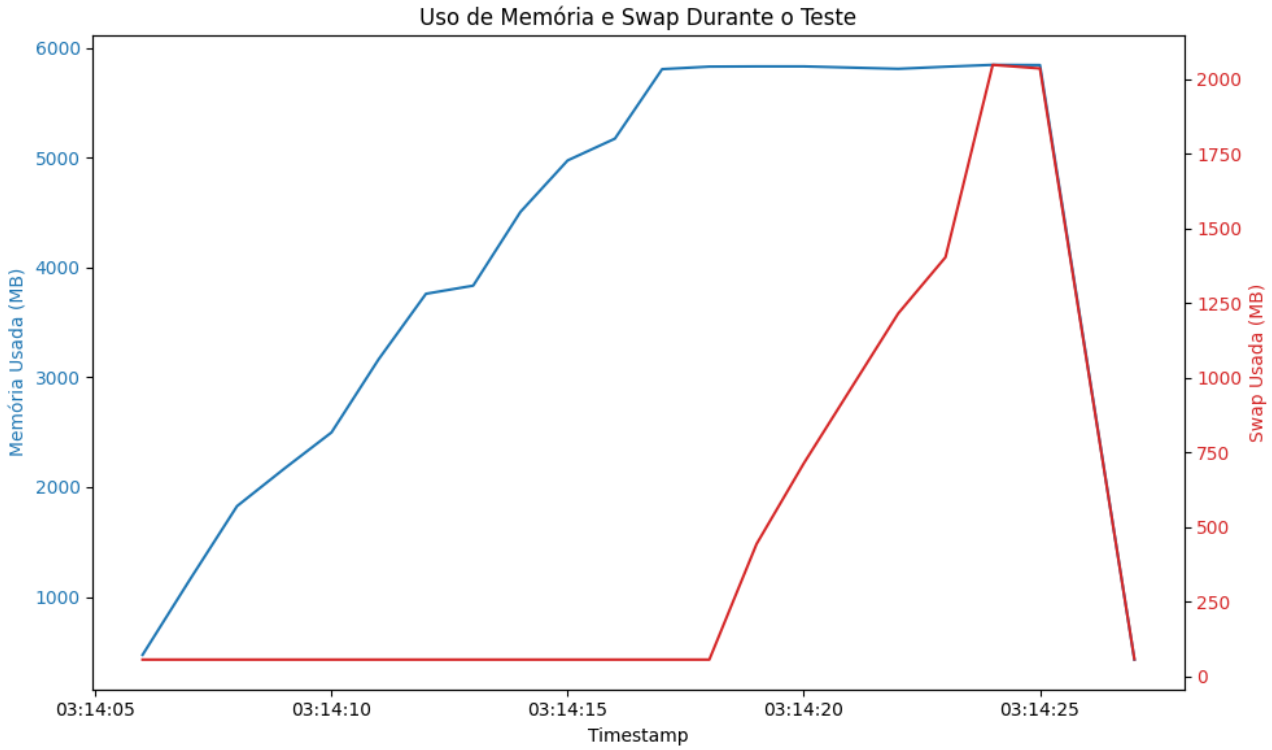
Teste 2 – Usando 500 MB/s durante 20s

```
(myenv) savioalves@Alves:~$ ./monitor_memory.sh
Tempo 1s - Memória alocada: 0.65 GB
Tempo 2s - Memória alocada: 1.30 GB
Tempo 3s - Memória alocada: 1.96 GB
Tempo 4s - Memória alocada: 2.61 GB
Tempo 5s - Memória alocada: 3.26 GB
Tempo 6s - Memória alocada: 3.91 GB
Tempo 7s - Memória alocada: 4.56 GB
Tempo 8s - Memória alocada: 5.22 GB
Tempo 9s - Memória alocada: 5.87 GB
Tempo 10s - Memória alocada: 6.52 GB
Tempo 11s - Memória alocada: 7.17 GB
./monitor_memory.sh: line 5: 10901 Killed                  python3 memory_overload.py
(myenv) savioalves@Alves:~$
```

Relátorio do Arquivo CSV - Teste 2

```
(myenv) savioalves@Alves:~$ cat memory_usage.csv
Timestamp,MemUsed,SwapUsed
03:14:06,473,57
03:14:07,1157,57
03:14:08,1827,57
03:14:09,2168,57
03:14:10,2498,57
03:14:11,3165,57
03:14:12,3760,57
03:14:13,3834,57
03:14:14,4505,57
03:14:15,4974,57
03:14:16,5172,57
03:14:17,5805,57
03:14:18,5828,57
03:14:19,5830,444
03:14:20,5830,714
03:14:22,5808,1216
03:14:23,5827,1404
03:14:24,5844,2048
03:14:25,5842,2036
03:14:27,430,58
(myenv) savioalves@Alves:~$
```

Gráfico – Teste 2



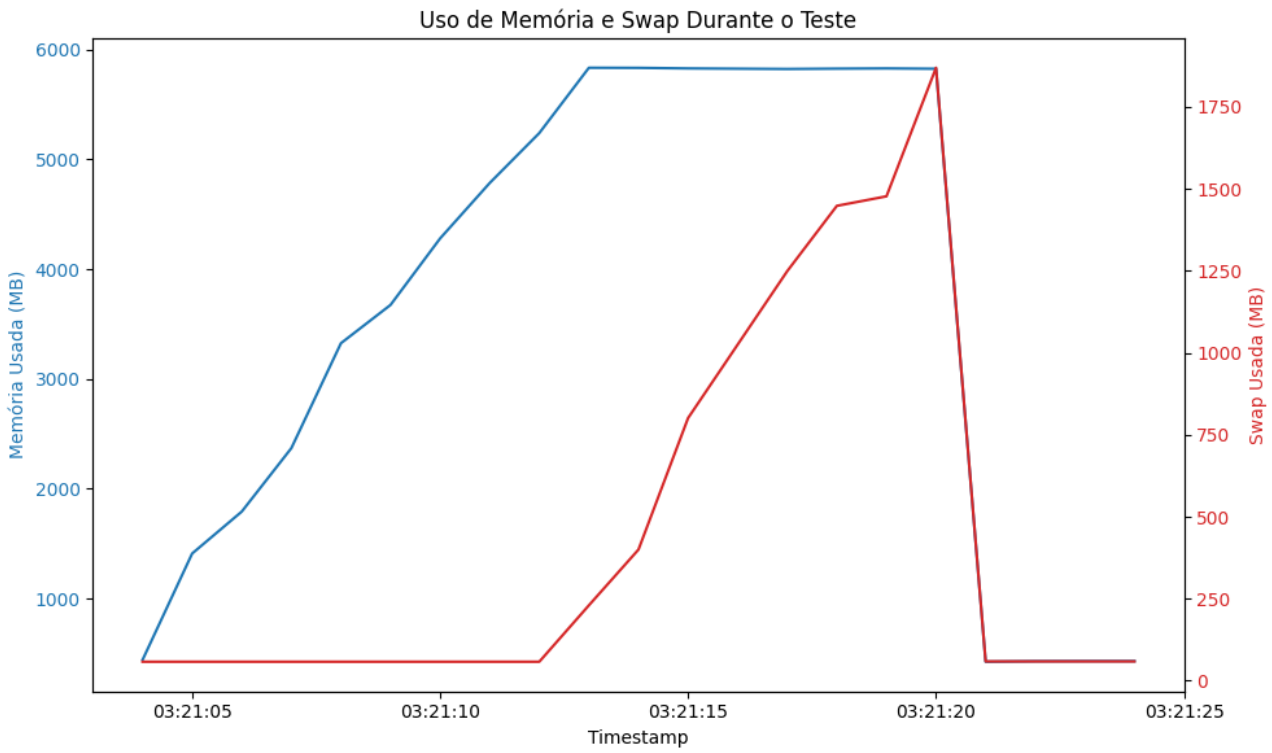
Teste 3 – Usando 900 MB/s durante 20s

```
(myenv) savioalves@Alves:~$ ./monitor_memory.sh
Tempo 1s - Memória alocada: 0.93 GB
Tempo 2s - Memória alocada: 1.86 GB
Tempo 3s - Memória alocada: 2.79 GB
Tempo 4s - Memória alocada: 3.73 GB
Tempo 5s - Memória alocada: 4.66 GB
Tempo 6s - Memória alocada: 5.59 GB
Tempo 7s - Memória alocada: 6.52 GB
./monitor_memory.sh: line 5: 11951 Killed                  python3 memory_overload.py
```

Relátorio do Arquivo CSV - Teste 3

```
(myenv) savioalves@Alves:~$ cat memory_usage.csv
Timestamp,MemUsed,SwapUsed
03:21:04,441,58
03:21:05,1410,58
03:21:06,1792,58
03:21:07,2370,58
03:21:08,3326,58
03:21:09,3675,58
03:21:10,4281,58
03:21:11,4786,58
03:21:12,5241,58
03:21:13,5836,230
03:21:14,5835,400
03:21:15,5830,801
03:21:17,5825,1249
03:21:18,5828,1448
03:21:19,5830,1477
03:21:20,5827,1869
03:21:21,424,59
03:21:22,429,59
03:21:23,430,59
03:21:24,430,59
```

Grafico – Teste 3



Questões para responder:

1. foi o Qual impacto do programa no uso da memória RAM e do swap?

- Teste 1
  - Impacto na RAM: O programa consumiu 1.86 GB de RAM após 20 segundos, aumentando o uso de memória de forma linear.

    - Impacto no SWAP: O programa foi executado sem erros, é houve RAM suficiente e o swap não tenha sido necessário.

- Teste 2
    - Impacto na RAM: O programa consumiu aproximadamente 0.65 GB/s. Consumindo a ram totalmente e forçou o sistema a utilizar o swap.
    - Impacto no SWAP: Forçado a utilizar o swap complentamente, o SO interrompe o programa com o "killed" para evitar instabilidade no sistema.
- Teste 3
    - O mesmo problema do Teste 2, o SO foi obrigado a interromper.

2. A swap foi utilizada? Em que momento isso ocorreu?

- Teste 2. a memória swap foi utilizada. Isso ocorreu a partir do timestamp 03:21:13, quando o valor de SwapUsed aumentou de 58 para 230. O uso da swap continuou a aumentar nos momentos seguintes, atingindo um pico de 1869 no timestamp 03:21:20. Após o timestamp 03:21:21, o uso da swap caiu drasticamente para 59, sugerindo que o programa que estava consumindo muita memória foi interrompido ou finalizado, liberando a memória RAM e reduzindo a necessidade de swap.

- No Teste 3 - Inicialmente, o uso de swap permanece constante em 57. A partir do timestamp 03:10:19, o uso de swap começa a aumentar, atingindo 44, 71 e, posteriormente, um pico de 2048 no timestamp 03:1U:24.Após o pico, o uso de swap cai drasticamente para 58 no timestamp 03:1U:27, coincidindo com a queda no uso de RAM.

3. O sistema operacional apresentou alguma lentidão durante o experimento?

- Não, porque o Sistema Operacional finalizou antes de preecher totalmente a RAM(12 GB) para questão de lentidão e segurança. Utilizando o OOM Killer (Out of Memory Killer)

4. O que acontece se o programa for executado com vm.swappiness=10 e vm.swappiness=100?

- No experimento com 12 GB de RAM, a configuração de vm.swappiness=10 faz com que o sistema priorize o uso da RAM e só utilize a swap quando a memória estiver quase esgotada. O programa continuará alocando memória na RAM até que ela esteja próxima do limite, e o uso de swap será mínimo. No entanto, se o consumo de memória for muito alto e o sistema não conseguir liberar RAM suficiente, o Linux pode ativar o OOM Killer para encerrar processos e evitar travamentos. Embora o desempenho geral permaneça alto, o sistema pode sofrer lentidão devido à pressão na RAM.

- Com vm.swappiness=100, o sistema passa a utilizar swap mais cedo, mesmo quando ainda há RAM disponível. Isso permite que a RAM seja liberada rapidamente para outros processos, mas a consequência é um maior uso da swap, o que pode impactar negativamente o desempenho, pois o acesso à swap é muito mais lento do que à RAM. O programa pode continuar rodando por mais tempo, já que o sistema evita o esgotamento completo da RAM, mas a performance pode ser prejudicada devido à maior dependência do disco.

Entrega do Trabalho

O documento final deve conter:
- Respostas às questões teóricas.
- Descrição da execução dos comandos e interpretação das saídas.
- Análise dos resultados do experimento prático.

Formato: PDF.

Data de entrega: 18/2/2025.

Local de entrega: Sigaa.